OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Sentiment Analysis: Transformers and Other Deep Learning Models

Faculty of Technology, Art and Design
24/05/2024

Authors

Reynato Jr. Matencio
Dung Thuy Vu
Jonas Lysfjord Kemi

Abstract

The present report showcases a project where a team of Master students trained and tested deep learning models using a large dataset containing millions of rows to classify product review sentiments in a positive or negative category. The students used several models to determine the best model based on accuracy with the hypotheses that a transformer model would be the best performer. Due to memory constraints on the computers utilized, all models except the pre-trained DistilBERT model used a sample size of 150 000 rows while the DistilBERT model was trained on 10 000 rows. Despite being trained on a fraction of the population, and less data than the other models, HuggingFace's DistilBERT model performance was superior on every metric measured. With an accuracy of 94% it beats the other transformer model built with TensorFlow with 4 more percentage points. CNN, RNN and LSTM all performed with accuracy below 90% supporting the case and the hypotheses that the transformer models regardless of the model being pre-trained were the superior for sentiment analysis.

Keywords: Sentiment, Analysis, Students, Review, NLP, Deep Learning, CNN, RNN, LSTM, Bi-LSTM, Transformer, BERT, DistilBERT, TensorFlow, Artificial Intelligence, Machine Learning

Contents

1 Introduction

The convenience and accessibility of online shopping have led to an explosion of user-generated reviews. These reviews offer a wealth of information, capturing consumer sentiment and preferences, making them a critical resource for businesses and researchers alike. With platforms like Amazon offering vast selections and the ability to leave reviews. These reviews have become a valuable source of information, influencing purchasing decisions and shaping product development. [Gope et al., 2022].

Nowadays, people prefer to shop online because of its convenience, wider selection, price comparison, reviews, and recommendations. Hence, these companies want to acquire and examine consumer feedback. [Haque et al., 2018].

In recent years, machine learning tools and models have conducted several sentiment analyses to examine consumers' opinions and feedback. As stated by [Chen and Sun, 2017] *"Analyzing and predicting consumer behavior has always been a blooming and promising area of study with great value of research"*. As a result, sentiment from the consumer plays a significant role in analyzing these reviews, assessing them, and assisting its users regarding this issue [Gope et al., 2022]. Additionally, obtaining insights from sentiment analysis has been a prerequisite for many big companies that allow consumers to review purchased items. This item review contains thousands of opinions from customers' feedback which holds valuable information about how a buyer's perceptions and reactions to its product [Huang et al., 2023]. That is why sentimental analysis combines natural language processing (NLP) and text analysis to potentially identify positive and negative opinions, emotions, or evaluations precisely [Chen and Sun, 2017].

1.1 Problem Statement

Understanding consumer sentiment can be challenging, despite the success of Amazon and its fully-monitored sentiments. A seeming escalation of Big data can trigger Amazon's sentiment analysis to conduct further studies when it comes to extracting and acquiring powerful insights from the data of reviews. Thanks to super-computer advancement, Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning have made a big contribution by conducting many research and survey articles associated with sentimental analysis. [Gope et al., 2022].

1.2 Project Scope

The primary aim of this project was to train and test different kinds of machine-learning models that could help the students compare and analyze distinct outcomes. Applying these machine-learning models would illuminate the students to acquire and deliver different results from the given datasets.

2 Theoretical Background

Sentiment analysis's main objective is to determine and classify the positive and negative sentiments expressed in a text. Nowadays, the emergence of big data in social media & e-commerce platforms researchers actively investigate and explore the importance of this analysis. [Gope et al., 2022].
Furthermore, [Tammina and Annareddy, 2020] stated that sentiment analysis could be utilized for recommending products according to the user's preferences, or predicting political election results that suit users' reviews of political parties gathered on Facebook, Twitter, and other social media platforms.
The essence of sentiment analysis is undeniably helpful, especially in the field of e-commerce. [Mittal et al., 2016] articulated that consumers these days tend to rely on social platforms whether they like it or not, as e-commerce constantly develops. Expressing the user's feedback, comments, and emotions takes a matter of minutes to generate via social media sites. Also, the authors highlighted that *"Sentiment analysis has uplifted the importance of both of these mentioned areas that are growing parallel with the growing globalization as the abundance of opinion data exists on multiple sites such as Facebook, Twitter, Amazon, etc. Emoticons are the new dramatic and pictorial format of conveying emotions in a written format"*.

Interestingly, sentiment analysis methodology is split into 3 categories: Below are the three hierarchical levels:

1. Document-level analysis: it simplifies the whole text documents in a singular component and determines the positive and negative sentiments as one theme.

2. Sentence level analysis: simplifying the feedback as one sentence and elaborating the positive and negative reviews of each sentence, sentence level analysis might be subjective or objective. Whilst in sentence level analysis a singular unit of expression is contemplated as a document.

3. Sub-sentence level analysis: this explains the favorable and unfavorable sentiments in the small group of words. Also, this could be subjective or objective. The singularity of the words is considered as a document as well [Chen et al., 2017]

2.1 Related to Work

To fully apprehend the true value of a product, the buyer must go through a thorough inspection of all the hundreds of reviews that have been posted by the previous before purchasing it. Given the fact that machine learning performs several powerful models to extract information from those sentiments, it also helps to provide a conclusion to identify what kind of sentiments it is. [Wedjdane et al., 2021].
The current state-of-the-art approach for sentiment analysis comprises lexicon-based, rule-based, machine learning, and deep-learning methods. These techniques enhance customer service, assist the marketing plan, foster the credibility of governance, deliver analytical data, and expand marketing endeavors [Huang et al., 2023]. Below are the following strategies that can be applied to the Amazon platform sentiments.

2.1.1 *Machine Learning*

The evolution of machine learning and its advanced models generates and provides us with rapid customer evaluations; if the model is deployed, it polarizes the sentiments and learns from them. One good example is applying a supervised learning model that could both train and filter the positive and negative feedback from the buyer. One poll conducted on Amazon's data over the past few years, found that the majority of the buyers provided positive feedback and expressed contentment. [Zhang, 2020].

2.1.2 *Lexicon-Based Approach*

One of the basic techniques for sentiment analysis is called the lexicon-based approach. The technique determines the entire document or a collection of sentences from the semantic orientation of lexicons. The subparts of semantic orientation normally divide into three categories; these are positive, negative, and neutral. [Gupta and Agrawal, 2020]

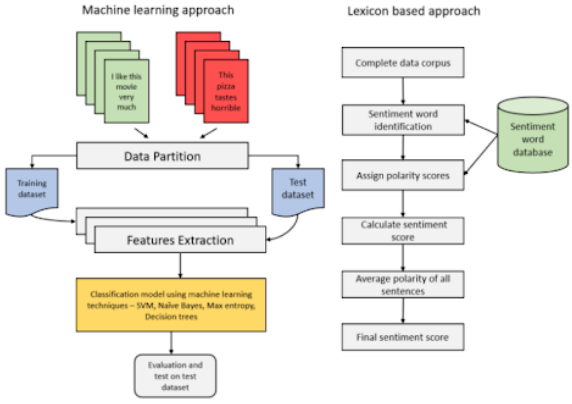

Figure 1: *Comparing the 2 types of Architecture between Lexicon & ML approach* [Tammina and Annareddy, 2020]

2.1.3 *Deep Learning Models*

Over the past few years, deep learning models considered one of the most accurate results when it comes to testing in sentimental analysis. The bag-of-word approach is one of the best examples showcasing the feature generation [Shrestha and Nasoz, 2019]. Implementing a systematic way of showing words in vector space, and an unsupervised technique was used to embed an unending vector space, whereas the given points were correlated to semantically related words [Mikolov et al., 2013].

1. ***Convolutional Neural Network (CNN) model in Sentiment Analysis***
   CNN is one of the well-known machine-learning models in the data revolution. It is popular because of its effectiveness when it comes to various experiments such as handwritten digits, facial recognition, and last but not least emotional recognition. One study introduced [Tammina and Annareddy, 2020]. has shown that in the new era of NLP, similar mechanisms for detecting features in image data sets at the same time to classify the photo that is employed can surpass conventional machine learning models if it is trained properly. The result shows that a more accurate tuning of hyper parameters such as filter size, number of kernels, and dropout probability could obtain a better extrapolation on the trained model.

2. ***Recurrent Neural Network model (RNN) in Sentiment Analysis***
   Another subset field of Artificial Neural Networks is Recurrent Neural Network which applies sequential data or time series data. Siri, Command Voice, and Google Translate are some of the useful programs that use RNN. This deep learning algorithm is famous for the use of ordinal or temporal problems such as language translation, speech recognition, and image detection [IBM, nd]. One study conducted by [Can et al.,] has a question *"Can a sentiment analysis model trained on a language be reused for sentiment analysis in other languages, Russian, Spanish, Turkish, and Dutch, where the data is more limited?"*. The experiment presented and suggested that the multilingual approach outperforms the lexicon-based benchmark as well as the overall benchmark.

3. ***Bidirectional Encoder Representations from Transformers (BERT) model in Sentiment Analysis***
   BERT or Bidirectional Encoder Representations from Transformers is one of the most popular deep learning models in NLP [Koroteev, 2021]. Although, we still do not totally comprehend the success on it [Rogers et al., 2021]. Technically, the main function of this model is for language understanding, feature extraction and fine-tuning for specific tasks. Fundamentally, pre-training and fine-tuning are the 2 main key steps for the standard BERT method. NSP which stands for sentence prediction analyze the two inputs in sentences from near to one to another, and MLM which means masked language model that predicts different masked input tokens [Rogers et al., 2021]. One study has shown that during the pandemic era from 2020, a lot of people expressed their frustration on social media platforms. Twitter was one of the popular social media platform compared to the other platforms [Singh et al., 2021]. The research used two data sets: one collected by tweets across the globe while the other data sets we selected only from the country of India. The findings indicate that understanding the mental state using their same eagerness and opinions illustrated the validation accuracy in approximately more or less 94% [Singh et al., 2021].

4. ***Long short-term memory (LSTM) model in Sentiment Analysis***
   The LSTM (long short-term memory) is a unique subset type of RNN recurrent neural network. Interestingly, one study showed a combine CNN and LSTM to forecast the emotion of the clients testimonials [Behera et al., 2021]. The final results outcome shows, in terms of accuracy as well additional characteristics or traits portrays ensemble model performs more effectively than conventional machine-learning models [Behera et al., 2021]. Another study proposed on text-based sentiment analysis using LSTM approached proved 85% accuracy by applying sentiment classification to run and train more data [Murthy et al., 2020].

5. ***Transformers in Sentiment Analysis***
   Transformers models holds a groundbreaking achievement in deep learning, especially in natural language processing articulated by [Vaswani et al., 2017] on their seminar paper called "Attention is all you need". Transformers can be defined as *"a neural network that learns the context of sequential data and generates new data out of it."* [Ferrer, uary]. In other words, transformers is an artificial intelligence model that comprehend and synthesize human-like text that helps analyzing patterns in a large amount of text data.

3 Research Methods

This section provide insight into the methods and models the students utilized to derive the results. The dataset was collected from an open source website named Kaggle. The next subsections will go through the process of data exploration, cleaning and transforming. The design of different models in deep learning such as CNN, RNN, BERT and Transformers. Lastly, the process of gathering the results to analyze the distinct outcome of each models.

3.1 Data Processing

The process of cleaning and acquiring the generated text for analysis is called pre-processing. Technically, texts that are found in social media contains a lot of background noise and useless elements like scripts and HTML tags. Moreover, a huge volume of text contains no impact for the overall trajectory. By maintaining the large amount of these words could lead to inaccuracies to the analytical procedure. [Mhatre et al., 2017]. Furthermore, data preparation process standardize the text data and remove the unnecessary parts or noise components [Tan et al., 2023].

Pandas & Numpy
Pandas or the Python Data Analysis Library is a versatile and well-known framework for data analytics [Hagedorn et al., 2021]. This innovate library is designed to facilitate vast amount of data easier and to supply the building blocks to put statistical model into practice. Numpy, on the other hand is the backbone library for scientific computing in Python. The abbreviation Numpy stands for *Numeration Python* is also considered as the cornerstone of numerous python modules. It is extensively utilized for scientific computing, especially for data analytics. It offers a high-performance function and data structured from the fundamental packages of python that are unable to deliver [Sapre and Vartak, 2020]. The figure 3 illustrates the difference between Pandas and Numpy.
The importance of using these two for sentiment analysis are for numerical calculation purposes and data manipulation. Without these, it would be impossible to cleaned, prep processed and organized data-frames for analysis and visualization. In addition, in order to process text embedding and deliver the complexity of mathematical computations that typically requires for machine learning models. [Majumder, 2021, Datagy, 2021].

3.1.1 *Stemming vs. Lemmatization*

The role of stemming in data prepossessing for analyzing the sentiment is pivotal. One study articulated that stemming is the procedure of mapping various word structure. It is also one of the basic technique of text pre-processing that used for various applications relation to information retrieval, natural language processing and language modelling [Jasmeet and Vishal, 2016]. In other words, stemming is a process in sentiment analysis to reduce the words to their root forms which can be clarified the accuracy of text classification.

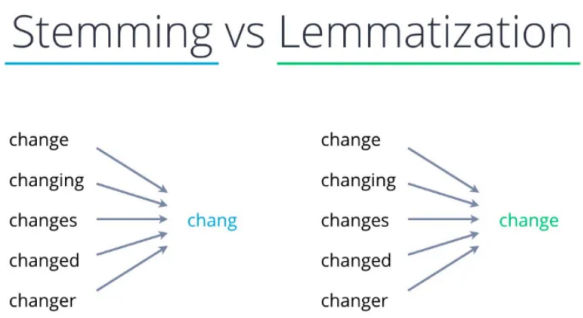

Figure 2: *Stemming VS. Lemmatization Models* [Bhoi, 2022]

Lemmatization
Lemmatization is the process of determining a normalized form of words [Plisson et al., 2004]. The primary goal of lemmatization is to reduce a single word to its root form, also known as *"lemma"*. For instance, a part of speech verb like word *"running"* would be identified as *"run"*. To put in another way, lemmatization is the study of the relationship between words, morphology, and its structure [Gillis, 2023].

3.1.2 *NLTK vs. spaCy*

NLTK and spaCy are the common tools for the developers, researchers and businesses. In terms of tasks such as tokenization, part of speech tagging, NLTK flexible algorithms, and spaCy is popular for its speed and efficiency that suits perfectly for NLP implementation. Although, both of this libraries are widely utilized in commercial application, NLTK serves researchers whereas spaCy handles production tasks [Limited, nd].

Aspect	NLTK	spaCy
Integration with other Python libraries	Offers seamless integration with NumPy, SciPy, pandas, and scikit-learn, facilitating data analysis	Compatible with pandas and NumPy for data handling; focuses on providing a self-contained solution
Compatibility with different operating systems	Written in pure Python, ensuring compatibility across Windows, macOS, and Linux	Leverages pure Python; provides pre-built binaries for various platforms
Support for multilingual NLP tasks	Built-in support for a wide range of languages, including English, French, German, Spanish, and more	Primarily focuses on English; support for other languages may require community contributions

Figure 3: *Comparative Analysis between NLTK and spaCy* [Limited, nd]

3.1.3 *Word Vector*

Word vector are one of the advanced techniques that is capable of examining the relationships between words, phrases and documents. The old-fashioned approach of representation of words has been replace due to the advancement of the technology by providing much more information. Thanks to word vector technologies like speech recognition and machine translation were developed [Ahire, 2022]. One study by [Ye et al., 2018] stated that to leverage the existing external knowledge in word vector for natural language processing classifies into 2 parts: *"1. Encoding External knowledge during the word vector learning stage. 2. Encoding external knowledge into pre-trained word vectors"*. With that being said, the first approach modifies the language model's objective function or supplement it with regularization words. On the other hand, the next approach modifies pre-trained words vectors by post-processing. As a results, it makes them lightweight. This means that these 2 methods work properly even though they contains various word vector type [Ye et al., 2018]. [Pennington et al., 2014].

The figure below depicts how the color code on each cells based on their value. Red if they fall in *positive* +2, white if they are fall in 0, and blue if they fall in *negative* -2.

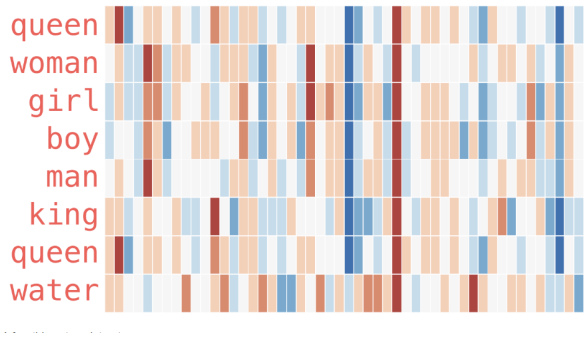

Figure 4: *Color Code Based Word Embedding* [Alammar, nd]

The image shows the different things to point out. 1. There is one straight red line for all the words. It simply means that they have a cohesive dimension along. 2. The word *woman* and *girl* are pretty the same in cells. Same as *man* and *boy*. 3. Apparently, the majority these words are person, while the word **water** is the distinct one. It indicates that the blue column goes all the way down and immediately stops before embedding for the word *water* [Alammar, nd].

Another good example for word embedding is the next-word prediction feature for a keyboard in our smartphone. Technically, next-word prediction is an application of NLP *Natural Language Processing* or another term for this is *Language Modelling*. Predicting the next word in a sentence is the purpose of language modelling [Soam and Thakur, 2022].

Figure 5: *Next-word Prediction* [Alammar, nd]

3.1.4 *Tokenization*

A process of dividing the words up into small pieces is called tokenizing. A single character, a phrase or even punctuation could be consider as token [Perry, 2022]. Furthermore, an article stated that tokenization *"Tokenization is the process of converting a string of text into a list of tokens (individual words/punctuation) and/or token IDs (integers that map a word to a vector representation of that word in an embedding array)"*. However, the definition of [Park et al., 2020] says that tokenization is the primary step in the most text processing operations. A token plays a significant impact on model's performance, because it serves as atomic unit that embeds text's contextual information. The picture illustrates how a simple tokenization happen.

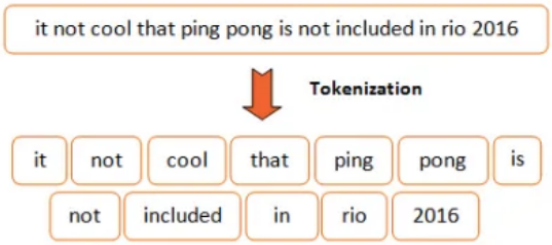

Figure 6: *Word Tokenization* [Felixmutai., 2023]

From a simple sentence that breaks down a string of text into units. Consequently, the sentence is narrowed down into following tokens: *["it", "not", "cool", "that", "ping", "pong", "is", "not", "included", "in", "rio", "2016"]*. This means that each cell is considered as separate token. In addition, several package/modules from NLP must import to obtain such results [Felixmutai., 2023].

3.1.5 *Pad sequences*

In natural language processing (NLP), pad sequences are normally applied and any other sequence-based to make sure where input sequence are in the same length. Typically, zeros or a token are either truncated long sequence or padded with a specific value [Srivatsavaya, 2023]. In the CNN operation particularly in feature map, padding is a method to keep the spatial dimensions of the input image. It adds an extra pixels to the input feature map boundaries [GeeksforGeeks, 2023].

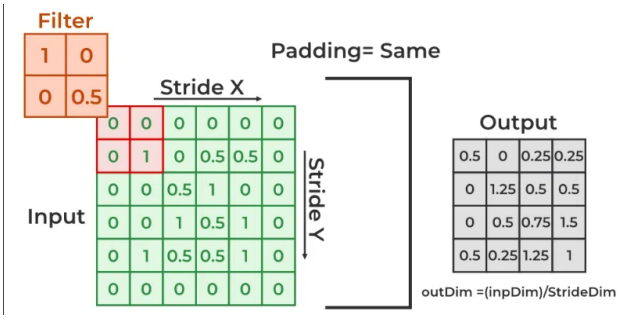

Figure 7: *Padding in CNN* [GeeksforGeeks, 2023]

The image demonstrates the process of adding layers on of zeros for the input to avoid some issues for the picture below.

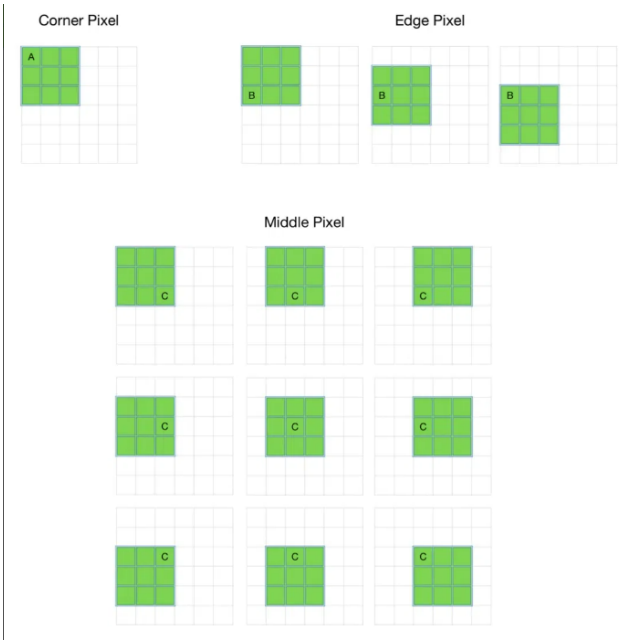

Figure 8: *Padding in CNN* [GeeksforGeeks, 2023]

The final output image is created and performed on a grey scale image of the dimension $n \times n$ with $f \times f$ filter. The figure indicates that the image gets smaller by each convolution. Hence, it restricts the depth of networks by limiting the amount of times the process can be performed before the size of the picture became zero. As a result, central pixels are more utilized frequently in comparison to the edge and corner of the pixels. [GeeksforGeeks, 2023].

3.1.6 *Word-embedding using pre-fraud model Word2Vec*

The technique that maps words to vectors of real integers is named word embedding. It shows how the words or phrases in multi-dimensional vector space [GeeksforGeeks, 2024]. Additionally, the Word2Vec is inspired from a free-forward fully connected architecture. For example, a simple sentence *"the quick brown fox jumped over the lazy dog"* and examine each word in the proper setting one by one. Technically, the words *"fox"* would rely on the context *"the quick brown"* if we apply forward context of size 3 and therefore the word *"jumped"* would depends on the context *"the brown fox"*, and so on [Silipo, 2018]. Word2Vec typically utilizes in two architectures:

Continuous Bag of Words (CBOW) This model forecasts the given context of words inside a particular window. The word of context are situated in input layer, meanwhile the output layer is the current word. Consequently, the number of parameters we want to represent is the current word that is now present at the input layer which contained in the hidden layer. [GeeksforGeeks, 2024].

Skip Gram This approach predicts the surrounding context words in a given window. Current word contained the input layer and the context words contained the output layer. Consequently, the number of parameters we want to represent is the current word that is now present at the input layer which contained in the hidden layer.

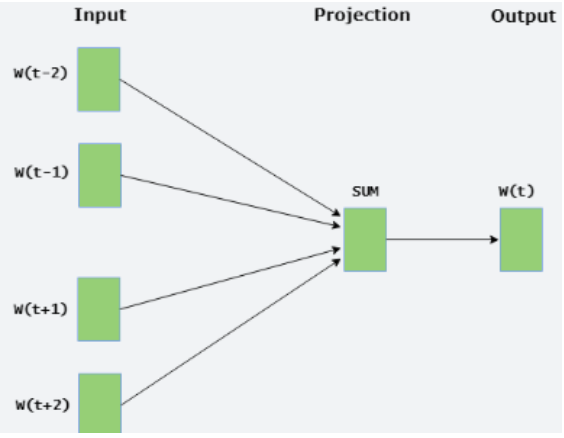

Figure 9: *CBOW-Continuous Bag of Words*

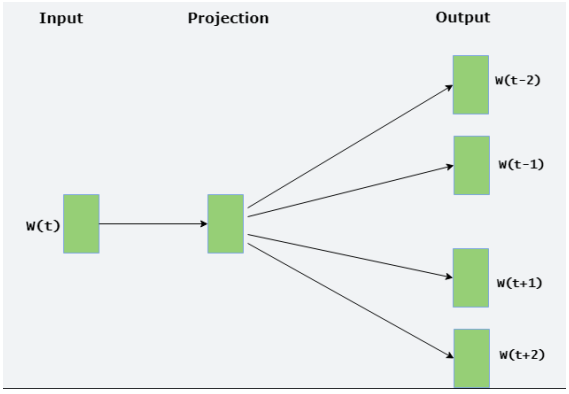

Figure 10: *Skip Gram*

3.1.7 *GloVe*

Global vectors for Word Representation or GloVe is designed to capture the semantic correlation between words [GeeksforGeeks., 2024]. A study of [Kamyab et al., 2021] stated that GloVe word embedding methods is extracting meaningful information from the textual data. The image below is one of the example of GloVe approach:

```
'the': [-0.123, 0.353, 0.652, -0.232]
'the' is very often used word in texts of any kind.
its equivalent 4-dimension dense vector has been given.
```

Figure 11: *GloVe Example* [GeeksforGeeks., 2024]

Academically, GloVe is consider as unsupervised learning algorithm to accomplish vectors representation for words that was developed by Stanford to produce word embedding by combining global word-to-word co-occurrence matrix from a corpus [Team, 2024] The GloVe comprises of pre-defined dense vectors for many common symbols such as braces, semicolons, and commas at the same time for every 6 billion words of English literature. Extensively, the algorithm's developers are constantly available for making the GloVe embedding accessible [GeeksforGeeks., 2024].

3.1.8 *TensorFlow*

TensorFlow is developed by the researchers at Google and it is known among the plethora of deep learning libraries [Pang et al., 2020]. It is pivotal for sentiment analysis for enhancing and creating an advanced deep learning learning models such as LSTM and CNN [Majumder, 2021]. Furthermore, it supports the GPU for deep learning to accelerate the workflow by leveraging the parallel processing capabilities of computations that leads in quicker and efficient results of deep learning workflows [Malingan, 2024].

3.1.9 *Hugging-Face*

In the world of AI community, hugging face plays a pivotal role that is known for its extensive range and equipment and libraries particularly in relation to natural language processing [Expedition, 2024]. It is considered a celebrity in the field of artificial intelligence, as it offers cutting-edge NLP tool and models. Technically, it is an arsenal that is designed to comprehend and create a human language. Similarly as its like a linguistic wizard. Furthermore, hugging-face is a well-liked paradigm as part of Transformers library. It is embedded with the APIs and tools that may quickly obtain and train with pre-trained models. [Shivanandhan, 2024].

4 Results

4.1 Data Cleaning

The text underwent a series of transformations including stop word removal, contractions handling, stemming, URL, and user mention replacement. The "contraction.csv" was utilized to expand contractions in the text data during the data cleaning process, ensuring that common contractions such as "ain't" (expanding to "is not") and "can't" (expanding to "cannot"), are correctly interpreted. Specific identifiers such as URLs and user mentions are replaced with placeholder tokens ("URL" and "user", respectively) to eliminate noise. Emoticons were standardized to enhance representation consistency. The crucial decision was made to utilize stemming over lemmatization, primarily due to its simplicity and effectiveness. While lemmatization offers a more sophisticated approach by considering word context and producing valid dictionary forms, it may overly generalize words, leading to a loss of important nuances and potentially affecting model performance adversely. Additionally, lemmatization datasets often require duplicate removal due to the potential for redundant representations when reducing words to their base form without context. In this specific case, stemming was preferred for its efficiency in retaining more word variations, aiding in better model training for sentiment analysis, where capturing nuanced sentiment differences is crucial. These steps collectively contribute to the reduction of noise, standardization of text representations, and improvement in data consistency.

4.2 Data Processing

Initially, the text data was tokenized using the Keras Tokenizer class, splitting it into individual words and creating a vocabulary of the most frequent 10,000 unique tokens. This strategic decision ensured that the model captured the majority of meaningful content while avoiding computational burden and noise from less frequent terms. Each text sequence was converted into a sequence of numerical indices based on this vocabulary. For example, the tokenized sequences before padding include samples such as [1642, 17, 911, 633, 315, 134, 592, 72, 21, 995, 27, 956, 592]. The maximum sequence length observed was 118 words, while the minimum was 0 (potentially indicating empty reviews or errors), and the median 33. These sequences were then padded to ensure uniform length, with zeros added to sequences shorter than the maximum length of 100, and longer sequences truncated to fit this length. The padded sequences for the training set have a shape of (120,000, 100), indicating 120,000 text samples each represented by a sequence of 100 tokens. Similarly, the test set had a shape of (30,000, 100), corresponding to 30,000 text samples.
Following tokenization and padding, word embedding were generated using pre-trained GloVe vectors. These embeddings provide dense 100-dimensional vector representations for words, generated by aggregating global word-word co-occurrence statistics from a large corpus, aimed to capture

semantic similarities. An embedding matrix was created, mapping each word index to its corresponding GloVe vector. If a word was not found in the GloVe vocabulary, it was assigned a vector from a semantically similar word or a zero vector if no suitable match exists. The embedding matrix has a shape of (10,001, 100), corresponding to 10,000 words plus an additional index for padding, each with a 100-dimensional vector. The vocabulary coverage using the GloVe model was 98.31%, indicating that most words in the dataset's vocabulary are present in the GloVe vocabulary. With high percentage coverage the decision to not conduct an experiment on the other GloVe model with 50, 200 and 300 word dimensions was made. An embedding layer, initialized with non-trainable pre-trained GloVe vectors, was defined to convert input sequences into dense vectors. This thorough preprocessing pipeline effectively transformed the textual data into a rich numerical format, preserving semantic information and preparing it for input into deep learning models.

Data Cleaning and Processing for Transformer

The Transformer model's data processing pipeline differs from that of CNN, LSTM, or Bi-LSTM models. Instead of tokenizing text into individual words, Transformers operate on subword units or byte pair encodings, enabling them to handle out-of-vocabulary words and capture morphological variations more effectively.

Another distinction lies in the use of a TextVectorization layer, which converts text into sequences of integers, rather than the direct word-to-vector mapping employed in other models. This layer affords greater flexibility in preprocessing and allows for custom standardization functions. Importantly, the TextVectorization layer is adapted to the training data before vectorization, ensuring that the vocabulary is dynamically learned and tailored to the specific dataset, potentially improving model performance.

Finally, the output sequences from the TextVectorization layer are fixed in length, typically determined by the maximum sequence length supported by the Transformer model architecture (e.g., 512). This consistency in input format facilitates batch processing and contributes to smoother training convergence.

When using the Transformer model, TensorFlow datasets present several advantages over traditional data structures like dataframes or lists. First, their seamless integration with the TensorFlow ecosystem streamlines data loading, preprocessing, and input into models, eliminating the need for additional conversion steps. Second, TensorFlow datasets excel in efficient memory management, allowing for the handling of large datasets that may not entirely fit in memory. By processing and loading data in batches, memory usage is reduced, enabling training on extensive datasets. Third, TensorFlow datasets support parallel data loading, accelerating data processing and training, particularly on systems with multiple CPU cores or GPUs. This parallelism minimizes input/output bottlenecks and optimizes data pipeline performance. Lastly, TensorFlow datasets provide robust APIs for defining data transformation pipelines, offering flexibility in preprocessing and data augmentation before feeding it into models. These pipelines seamlessly integrate operations like standardization, augmentation, shuffling, and batching within TensorFlow.

4.3 Model Architecture and Accuracy

4.3.1 *Machine Learning model*

The initial step involved processing the textual data using the TfidfVectorizer to transform it into TF-IDF feature vectors. This approach was chosen over one-hot encoding due to its ability to capture not only the presence of words but also their relative importance within the corpus. Subsequently, a logistic regression model was implemented, chosen for its simplicity efficiency to compare it to the more complex deep learning architectures. The model was configured with a maximum of 100 iterations to ensure convergence. Given the context of a relatively small dataset of 150,000 Amazon reviews in our project and a binary sentiment analysis task of simply classify whether the review is positive or negative, it's not uncommon for simple machine learning techniques like logistic regression to perform comparably or even better than deep learning models such as CNN and LSTM.

With a limited amount of data, deep learning models are prone to overfitting, as evidenced by their performance degradation over 30 epochs(shown in the images of test accuracy trend line). Logistic regression, on the other hand, is less complex and less prone to overfitting, making it more robust in such scenarios. Additionally, the observed ML accuracy being 88% higher than CNN and almost equal to LSTM underscores the effectiveness of the simpler approach.

For the further analysis out of the scope of these projects with a bigger dataset , deep learning models could offer advantages in more complex tasks that require capturing nuanced semantic relationships or handling unstructured data formats beyond text such as analyzing sentiment in images or videos for instance.

Additionally, deep learning models have shown promising results in transfer learning, where pre-trained models can be fine-tuned on specific sentiment analysis tasks, potentially leading to improved performance even with smaller datasets . This scenario has been conducted by using DistilBERT Transformer model from Hugging face at the later part of the project.

4.3.2 *CNN*

Two distinct architectures were investigated here. The first model adopts a multi-layered structure, commencing with an input layer configured to handle sequences of word embedding with a length at 100. Subsequently, an embedding layer transforms input word indices into dense vectors of fixed size. Convolutional layers (Conv1D) followed. Each convolutional layer was paired with max-pooling operations, which serve to distill the most salient features. A global max-pooling layer was employed to extract the most pertinent features from each feature map across the entire sequence, reducing spatial dimensionality while retaining essential information. The architecture further includes dense layers for processing the extracted features, culminating in a final output layer for binary classification using the sigmoid activation function.

The second model architecture closely resembles the first, with notable differences in the convolutional layer configurations. While the first model features three convolutional layers with 128 filters of size 3, the second model incorporates a different arrangement, starting with 128 filters of size 5 followed by 64 filters of size 3. Additionally, the second model omits intermediate dense layers, comprised only two dense layers before the final output layer.

Model 1 exhibits overfitting tendencies, as evident from fluctuating validation accuracy and a gradual decrease over epochs. As an observation the second model offers a streamlined architecture aimed at balancing expressiveness and generalization capability and indeed smoother the accuracy and loss line of validation dataset which was a good implementation over model 1.

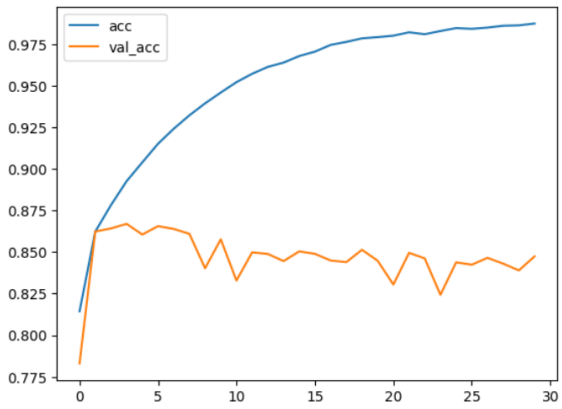

Figure 12: Graph 1: CNN Version 1

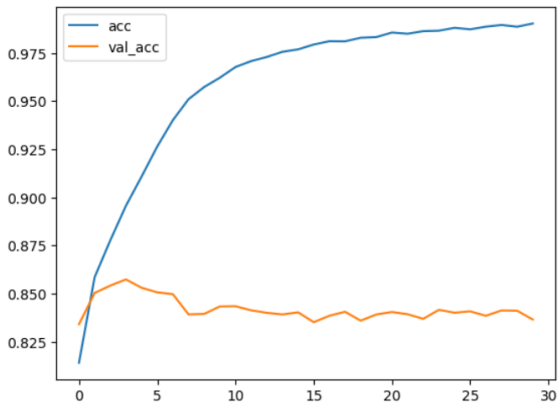

Figure 13: Graph 2: CNN Version 2

4.3.3 *LSTM*

The architecture of the LSTM model begins with an input layer configured to handle sequences of word embeddings. Following the embedding layer, two LSTM layers with 15 units each were employed, with the return_sequences=True parameter setting to enable the output sequences to be utilized by subsequent layers. Dropout regularization with a rate of 0.2 was incorporated between the LSTM layers to mitigate overfitting by randomly dropping a fraction of input units during training. A global max-pooling layer was then applied to condense the temporal sequence data into a fixed-length representation by selecting the maximum value over time for each feature. The output from the pooling layer was subsequently fed into dense layers for further feature extraction and abstraction, culminating in a final output layer with a sigmoid activation function for binary classification as sentiment analysis in this case.

The model was trained using the binary cross-entropy loss function and the Adam optimizer with a learning rate of 0.01. The model's performance was evaluated on the test dataset, achieving a test accuracy of approximately 87.58%. In comparison to the CNN model, the LSTM architecture tends to outperform in terms of accuracy. From the experiment, the LSTM model showed an ability to capture long-term dependencies within sequential data, such as text, was particularly advantageous in sentiment analysis tasks where the context of words plays a crucial role in determining sentiment. By virtue of its recurrent nature, the LSTM model excels at retaining memory over extended sequences, allowing it to better discern nuanced patterns and contextual cues that contribute to sentiment. This capability enables the LSTM to effectively extract and leverage semantic information from text, resulting in more accurate sentiment predictions.

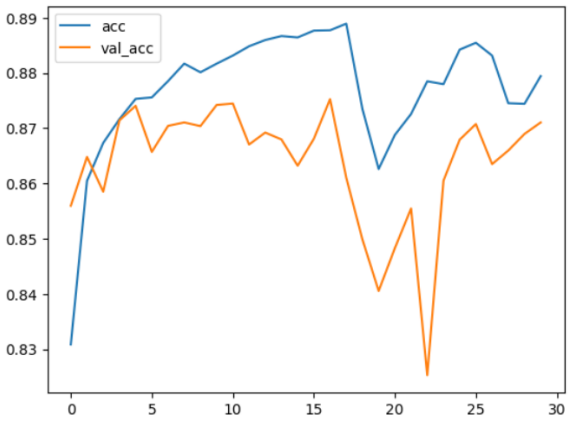

Figure 14: Graph 3: LSTM

4.3.4 *Bidirectional-LSTM*

The hypothesis that a Bidirectional LSTM (Bi-LSTM) would outperform a standard LSTM in sentiment analysis was based on its theoretical ability to capture contextual information from both past and future states within a sequence. While the Bi-LSTM did not yield a significant increase in overall accuracy compared to the standard LSTM (both achieving around 87% test accuracy), it did exhibit a notably smoother learning curve during training, suggesting more stable convergence. This smoother trajectory may be attributed to the Bi-LSTM's bidirectional nature, enabling it to better utilize contextual information and potentially mitigate the impact of noisy data points.
However, several factors could explain why the Bi-LSTM did not demonstrate a substantial accuracy improvement. First, the increased model complexity introduces more parameters, making it prone to overfitting, particularly with a limited training dataset of 150,000 reviews. Second, the quality and diversity of the Amazon review dataset could be a limiting factor. If the data lacks sufficient variety in sentiment expressions, the benefits of the Bi-LSTM's enhanced contextual understanding may not be fully realized. Third, hyperparameter tuning is critical for deep learning models. The Bi-LSTM may require different optimal hyperparameter settings than the standard LSTM to reach its full potential.

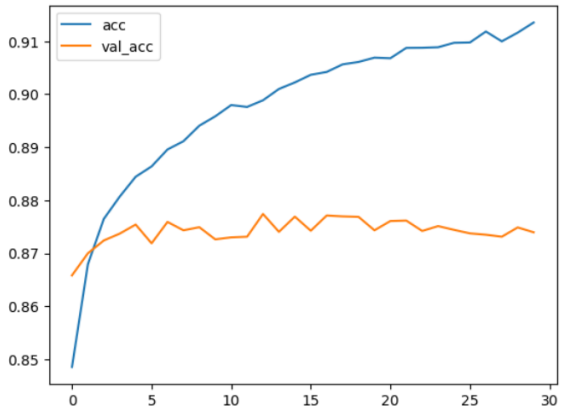

Figure 15: Graph 4: Bi-LSTM

4.3.5 *Transformers (TensorFlow)*

The Transformer model architecture used for sentiment analysis was built upon several key components, each contributing to its ability to effectively process and understand text data. The positional_encoding function generates embeddings that encode the positions of words within the input sequences, crucial for the Transformer model, which, unlike RNN or LSTM that inherently process data sequentially, processes all words in a sequence simultaneously and lacks an inherent understanding of word order. This function uses sinusoidal functions of different frequencies to ensure each position has a unique representation. These encodings are then added to the token embeddings, helping the model understand the relative positions of words in the sequence.
The Embedding layer combines token embeddings and positional encoding to form the input embeddings for the Transformer. Token embeddings convert each word in the sequence into a dense vector representation, capturing semantic information, while positional encodings are added to incorporate positional information. This combination allows the model to grasp both the meaning of words and their order in the sequence, which is essential for tasks like sentiment analysis.
The Transformer Encoder layer includes several important components: Multi-Head Attention, which enables the model to focus on different parts of the input sequence simultaneously, capturing dependencies between words regardless of their distance in the sequence; a Feed-Forward Neural Network, introducing non-linearity and further processing the information after attention; Layer Normalization, which stabilizes and accelerates training by normalizing the outputs of the previous layer; and Residual Connections, which help mitigate the vanishing gradient problem, ensuring effective gradient flow through the network. The model was constructed by stacking the embeddings layer and multiple transformer encoder layers, with the output flattened and passed through a dense layer with a sigmoid activation function to produce the final binary classification output.
The Transformer model achieved a test accuracy of 90.42%, demonstrating its effectiveness in the sentiment analysis task. This accuracy metric reflects the model's ability to correctly classify sentiment labels in the test dataset, with improvements across epochs indicating effective learning from the training data. The slight dip in performance towards the end suggests potential overfitting(from the graph).

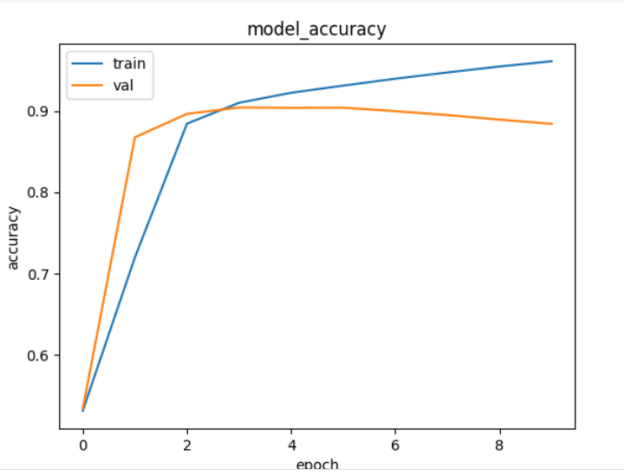

Figure 16: Graph 5: Transformer TensorFlow

4.3.6 *Transformers (HuggingFace)*

The DistilBERT model has been trained on a large corpus of text, which typically includes datasets such as Wikipedia and BooksCorpus. While the exact number of training examples may vary, it has been pre-trained on around tens to hundreds of millions of text samples to learn general language patterns and semantics. In terms of parameters, DistilBERT has approximately 66 million to 82 million parameters. The DistilBERT has about 40% fewer parameters than BERT, which makes it much faster and less resource-intensive while maintaining approximately 97% of BERT's performance. It achieves this by reducing the number of layers from 12 to 6. In terms of the depth of the model architecture. DistilBERT incorporates multiple layers of transformer blocks, providing deeper representations and potentially capturing more intricate patterns in the data, while the custom-built Transformer is more streamlined with a single layer of encoder, making it simpler and easier to customize for specific tasks.
This fine_tuning model was especially advantageous when the task-specific dataset was relatively small.The pre-trained model already has a robust understanding of language, which includes understanding sentiment-laden phrases, idioms, and context-specific word meanings, so it requires fewer task-specific examples to achieve high performance.In this case we chose pretty small dataset of 10000 reviews to achieve 94 % of test accuracy. During fine-tuning, the model adjusts its weights, aka minimize the loss function, to better fit the sentiment analysis task, specializing its understanding of how positive, negative, and neutral sentiments are expressed in text. This results in improved accuracy and performance compared to a model trained from scratch on the same dataset from 90% to around 94%.

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	No log	0.177882	0.935000	0.934970
2	0.165400	0.249237	0.932667	0.932614
3	0.111200	0.321766	0.930000	0.929962
4	0.049500	0.365720	0.930667	0.930665
5	0.040100	0.443852	0.929333	0.929260
6	0.013000	0.429985	0.938333	0.938323
7	0.010300	0.579970	0.927000	0.926880
8	0.010100	0.502068	0.934333	0.934335
9	0.010100	0.585580	0.930333	0.930233
10	0.011900	0.515840	0.933000	0.932994
11	0.011800	0.509370	0.933333	0.933334
12	0.009600	0.576314	0.929333	0.929321
13	0.009200	0.557845	0.930667	0.930669
14	0.011000	0.530525	0.933000	0.932969
15	0.005700	0.597613	0.927333	0.927238
16	0.004300	0.562080	0.936667	0.936663
17	0.004300	0.608210	0.936000	0.936002
18	0.001900	0.614418	0.935333	0.935334
19	0.000900	0.660776	0.935667	0.935658

Figure 17: Graph 6: HuggingFace DistilBERT

4.4 Discussion

4.4.1 *Implications*

Overfitting which is indicated by a decrease, or fluctuation in validation accuracy after a few epochs can be solved by several techniques:

1 **Regularization:** Techniques such as dropout (randomly dropping units during training) can prevent the model from becoming too reliant on specific neurons, promoting generalization. This involves trying a different range of drop out values.

2 **Early Stopping:** Monitoring the validation loss and stopping training when it stops improving can prevent overfitting within specific number of epochs such as 3 or 4

3 **Data Augmentation:** Expanding the training dataset with augmented data can improve model robustness.

Memory Constraints was an issue prevalent throughout the project. Given the large dataset of 3.6 million texts, we faced memory constraints for both the GPU and RAM and had to limit the data to 150,000 rows. This limitation can affect the model's performance due to insufficient training data. To address this, we could have implemented the follow methods:

1 **Data Generators:** Implementing a data generator can help manage memory usage by loading data in batches and only storing necessary parts in memory at any given time. In process, a data generator is a Python generator function that yields batches of data during training. Instead of loading the entire dataset into memory, a data generator loads a batch of data, processes it, and then yields it to the training process. This process repeats until all data samples have been processed. By loading data in batches, data generators help manage memory usage, as only a small portion of the dataset needs to be stored in memory at any given time. This allows the entire dataset to be utilized without overwhelming the system's memory. In sentiment analysis, data generators can efficiently handle large text datasets by loading and processing data on-the-fly, enabling the model to train on the entire dataset without overwhelming system memory.

2 **Dask DataFrames:** Dask is a parallel computing library in Python that enables efficient processing of large datasets by paralleling operations and splitting data into smaller, manageable chunks. In sentiment analysis, Dask DataFrames can be used to handle large text datasets by splitting them into smaller partitions, each of which can be processed independently. This parallelization allows multiple computations to be performed simultaneously, reducing the computational load and enabling efficient processing of the entire dataset. By leveraging Dask DataFrames, sentiment analysis models can train on larger portions of the dataset without encountering memory issues, leading to improved model performance and generalization.

3 **Downcasting numeric columns in pandas** This can be a helpful technique to reduce memory usage when working with large datasets. Downcasting numeric columns involves converting the data type of numeric columns to smaller, more memory-efficient types while ensuring that the data remains accurate and usable for analysis. For example, converting integer columns to smaller integer types or floating-point columns to lower precision floating-point types.

While downcasting numeric columns can certainly help reduce memory usage, especially for datasets with a large number of numeric features, it may not be sufficient to handle extremely large text datasets in this sentiment analysis tasks of 3.6 millions text review. Text data often dominates memory usage due to its variable length and complexity, and downcasting numeric columns alone may not adequately address this issue. In this case, The combination of all 3 tactics can be helpful.

4 **Difference metrics of model performance** Given dataset labels of 150000 samples, the class distribution was relatively balanced:

- Label 1: 76,248 samples
- Label 0: 73,752 samples

This results in a class distribution of approximately 50.8% for Label 1 and 49.2% for Label 0. This was fairly balanced, so significant class imbalance issues are unlikely to be a major concern. The decision to only use accuracy as model performance criteria was made for that reason. Further, other metrics such as precision and recall could be considered as thorough interpretation. For example , if precision was high but recall was low, it suggests that while the model correctly identifies positive sentiments most of the time, it misses many actual positive instances. Conversely, if recall is high but precision is low, the model captures a large portion of positive sentiments but also classifies many negative instances as positive. Balancing precision and recall allows for fine-tuning the model to prioritize specific aspects of sentiment detection, such as reducing false positives or capturing more positive sentiments. This tailored approach enhances the overall effectiveness of sentiment analysis by providing more accurate insights into sentiment classification.

5 Conclusion

In conclusion, this sentiment analysis project investigated a range of model architectures, each showcasing distinct strengths and weaknesses. While logistic regression provided a basic foundation, deep learning models like CNN, LSTM, and Bi-LSTM demonstrated superior accuracy due to their capacity to discern complex patterns within textual data. Notably, the Transformer model, especially when leveraging HuggingFace's DistilBERT, achieved the highest accuracy. This success can be attributed to its pre-training on extensive text corpora and efficient fine-tuning with a smaller dataset.

The project encountered challenges, particularly with memory limitations, necessitating the use of a smaller sample size of 150,000 reviews for most models. This constraint may have impacted the models' full potential, highlighting the importance of resource management in such analyses. Notably, DistilBERT's efficiency allowed for effective training on just 10,000 samples, underscoring its suitability for scenarios with limited data.
While accuracy served as the primary evaluation metric due to a balanced dataset, future work could incorporate precision and recall for a more nuanced assessment, particularly when dealing with imbalanced classes. Overall, this project demonstrated the efficacy of deep learning, particularly Transformer models like DistilBERT, in sentiment analysis tasks. It emphasized the significance of choosing appropriate architectures, pre-training, and data sampling strategies for achieving optimal performance.

Moving forward, exploring larger datasets if resources permit, conducting further hyperparameter tuning, and experimenting with ensemble methods could potentially enhance accuracy. Additionally, investigating the impact of different sample sizes on model performance would offer valuable insights into the trade-off between accuracy and computational resources.

References

[Ahire, 2022] Ahire, J. B. (2022). Introduction to word vectors. https://nirajbhoi.medium.com/stemming-vs-lemmatization-in-nlp-efc280d4e845. Accessed: 2024-05-15.

[Alammar, nd] Alammar, J. (n.d.). The illustrated word2vec. https://jalammar.github.io/illustrated-word2vec/.

[Behera et al., 2021] Behera, R. K., Jena, M., Rath, S. K., and Misra, S. (2021). Co-lstm: Convolutional lstm model for sentiment analysis in social big data. *Information Processing & Management*, 58(1):102435.

[Bhoi, 2022] Bhoi, N. (2022). Stemming vs lemmatization in nlp.

[Can et al.,] Can, E., Ezen-Can, A., and Can, F. Multilingual sentiment analysis: an rnn-based framework for limited data (2018). *arXiv preprint arXiv:1806.04511*.

[Chen and Sun, 2017] Chen, M. and Sun, Y. (2017). Sentimental analysis with amazon review data.

[Chen et al., 2017] Chen, T., Xu, R., He, Y., and Wang, X. (2017). Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

[Datagy, 2021] Datagy (2021). Pandas numpy tutorials. https://datagy.io/pandas-tutorials/.

[Expedition, 2024] Expedition, M. L. (2024). Hugging face transformers for sentiment analysis: A practical guide. https://www.machinelearningexpedition.com/hugging-face-transformers-for-sentiment-analysis-a-practical-guide/.

[Felixmutai., 2023] Felixmutai. (2023). Tokenization in sentiment analysis.

[Ferrer, uary] Ferrer, J. (2024, January). How transformers work: A detailed exploration of transformer architecture. *Explore the architecture of Transformers, the models that have revolutionized data handling through self-attention mechanisms.*

[GeeksforGeeks, 2023] GeeksforGeeks (2023). Cnn introduction to padding. https://www.geeksforgeeks.org/cnn-introduction-to-padding/.

[GeeksforGeeks., 2024] GeeksforGeeks. (2024). Pre-trained word embedding using glove in nlp models. https://www.geeksforgeeks.org/pre-trained-word-embedding-using-glove-in-nlp-models/.

[GeeksforGeeks, 2024] GeeksforGeeks (2024). Word embedding using word2vec. https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/.

[Gillis, 2023] Gillis, A. S. (2023). Lemmatization. enterprise ai. https://www.techtarget.com/searchenterpriseai/definition/lemmatization#:~:text=The%20goal%20of%20lemmatization%20is,and%20contextual%20analysis%20of%20word.

[Gope et al., 2022] Gope, J. C., Tabassum, T., Mabrur, M. M., Yu, K., and Arifuzzaman, M. (2022). Sentiment analysis of amazon product reviews using machine learning and deep learning models. In *2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, pages 1–6. IEEE.

[Gupta and Agrawal, 2020] Gupta, N. and Agrawal, R. (2020). Application and techniques of opinion mining. In *Hybrid computational intelligence*, pages 1–23. Elsevier.

[Hagedorn et al., 2021] Hagedorn, S., Kläbe, S., and Sattler, K.-U. (2021). Putting pandas in a box. In *Conference on Innovative Data Systems Research (CIDR);(Online)*, page 15.

[Haque et al., 2018] Haque, T. U., Saber, N. N., and Shah, F. M. (2018). Sentiment analysis on large scale amazon product reviews. In *2018 IEEE international conference on innovative research and development (ICIRD)*, pages 1–6. IEEE.

[Huang et al., 2023] Huang, H., Asemi, A., and Mustafa, M. B. (2023). Sentiment analysis in e-commerce platforms: A review of current techniques and future directions. *IEEE Access*.

[IBM, nd] IBM (n.d.). What are recurrent neural networks?

[Jasmeet and Vishal, 2016] Jasmeet, S. and Vishal, G. (2016). Text stemming: Approaches, applications, and challenges. *ACM Comput. Surv*, 49(3):1–46.

[Kamyab et al., 2021] Kamyab, M., Liu, G., and Adjeisah, M. (2021). Attention-based cnn and bi-lstm model based on tf-idf and glove word embedding for sentiment analysis. *Applied Sciences*, 11(23):11255.

[Koroteev, 2021] Koroteev, M. (2021). Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

[Limited, nd] Limited, S. T. P. (n.d.). Nltk vs spacy: A deeper dive into nlp libraries.

[Majumder, 2021] Majumder, P. (2021). Web scraping a news article and performing sentiment analysis using nlp. https://datagy.io/pandas-tutorials/.

[Malingan, 2024] Malingan, N. (2024). Accelerating deep learning with tensorflow gpu. https://www.scaler.com/topics/tensorflow/gpus-for-deep-learning/.

[Mhatre et al., 2017] Mhatre, M., Phondekar, D., Kadam, P., Chawathe, A., and Ghag, K. (2017). Dimensionality reduction for sentiment analysis using pre-processing techniques. In *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, pages 16–21. IEEE.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mittal et al., 2016] Mittal, S., Goel, A., and Jain, R. (2016). Sentiment analysis of e-commerce and social networking sites. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2300–2305. IEEE.

[Murthy et al., 2020] Murthy, G., Allu, S. R., Andhavarapu, B., Bagadi, M., and Belusonti, M. (2020). Text based sentiment analysis using lstm. *Int. J. Eng. Res. Tech. Res*, 9(05).

[Pang et al., 2020] Pang, B., Nijkamp, E., and Wu, Y. N. (2020). Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45(2):227–248.

[Park et al., 2020] Park, K., Lee, J., Jang, S., and Jung, D. (2020). An empirical study of tokenization strategies for various korean nlp tasks. *arXiv preprint arXiv:2010.02534*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Perry, 2022] Perry, T. (2022). What is tokenization in natural language processing (nlp)? https://www.machinelearningplus.com/nlp/what-is-tokenization-in-natural-language-processing/?utm_content=cmp-true.

[Plisson et al., 2004] Plisson, J., Lavrac, N., Mladenic, D., et al. (2004). A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86. Citeseer.

[Rogers et al., 2021] Rogers, A., Kovaleva, O., and Rumshisky, A. (2021). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

[Sapre and Vartak, 2020] Sapre, A. and Vartak, S. (2020). Scientific computing and data analysis using numpy and pandas. *International Research Journal of Engineering and Technology*, 7:1334–1346.

[Shivanandhan, 2024] Shivanandhan, M. M. (2024). How to build a simple sentiment analyzer using hugging face transformer. https://www.linkedin.com/pulse/how-build-simple-sentiment-analyzer-using-hugging-m-shivanandhan-wuidc/.

[Shrestha and Nasoz, 2019] Shrestha, N. and Nasoz, F. (2019). Deep learning sentiment analysis of amazon. com reviews and ratings. *arXiv preprint arXiv:1904.04096*.

[Silipo, 2018] Silipo, R. (2018). Word embedding: Word2vec explained. https://dzone.com/articles/word-embedding-word2vec-explained.

[Singh et al., 2021] Singh, M., Jakhar, A. K., and Pandey, S. (2021). Sentiment analysis on the impact of coronavirus in social life using the bert model. *Social Network Analysis and Mining*, 11(1):33.

[Soam and Thakur, 2022] Soam, M. and Thakur, S. (2022). Next word prediction using deep learning: A comparative study. In *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 653–658. IEEE.

[Srivatsavaya, 2023] Srivatsavaya, P. (2023). Advantages and disadvantages of pad_sequences — keras layer. *Medium*. Available at: https://medium.com/@prudhviraju/advantages-and-disadvantages-of-pad_sequences-keras-layer-xxxxxx.

[Tammina and Annareddy, 2020] Tammina, S. and Annareddy, S. (2020). Sentiment analysis on customer reviews using convolutional neural network. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE.

[Tan et al., 2023] Tan, K. L., Lee, C. P., and Lim, K. M. (2023). A survey of sentiment analysis: Approaches, datasets, and future research. *Applied Sciences*, 13(7):4550.

[Team, 2024] Team, G. L. (2024). What is word embedding — word2vec — glove. https://www.mygreatlearning.com/blog/word-embedding/.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wedjdane et al., 2021] Wedjdane, N., Khaled, R., and Okba, K. (2021). Better decision making with sentiment analysis of amazon reviews. In *2021 International Conference on Information Systems and Advanced Technologies (ICISAT)*, pages 1–7. IEEE.

[Ye et al., 2018] Ye, Z., Li, F., and Baldwin, T. (2018). Encoding sentiment information into word vectors for sentiment analysis. In *Proceedings of the 27th international conference on computational linguistics*, pages 997–1007.

[Zhang, 2020] Zhang, X.-D. (2020). A matrix algebra approach to artificial intelligence.